Amendments to the Claims:

This listing of the claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. **(Currently Amended)** A method for controlling access to data handled by references in a system for executing programs, said programs including processes and tasks, wherein upon executing a program, the method comprises the following steps:

- having the system store an entire set of references which the program obtains by means considered as licit, said licit reference being stored when introducing into the program said reference by a licit means and when this licit reference is not already stored~~said program comprising code from a single Java Card package~~;

- before any operation intended to be forbidden in case said operation deals with values which are not licit references, having the system check that said values are among the licit references which have been stored for this program, and

- accepting the operation, responsive to said step of checking, when said checking determines said values are among the licit references, and rejecting the operation responsive to said step of checking, when said checking determines said values are not among the licit references.

2. **(Previously Presented)**The method according to claim 1, wherein the references are pointers.

3. **(Previously Presented)** The method according to claim 1, wherein the licit means for a program in order to obtain reference values comprise at least one of the following operations:

- reading a variable or a datum belonging to the system or to another program,
- writing into a variable or datum of said program by the system or by another program,
- receiving arguments upon calling a routine of said program by the system or by another program,
- utilization of the return value from the call by said program of a routine belonging to the system or to another program,
- having said program catch up a raised exception during execution of a routine belonging to the system or to another program,
- receiving by said program an interruption or a valuated signal.

4. **(Previously Presented)** The method according to claim 1, wherein the system comprises a mechanism which determines whether a given value is a valid reference

.

5. **(Previously Presented)** The method according to claim 4, wherein the system comprises a firewall which forbids certain programs on certain referenced data, data considered as being sensitive for the system being those for which the operations are not forbidden by the firewall.

6. **(Previously Presented)** The method according to claim 5, wherein the firewall forbids certain operations by a program on data belonging to other programs, except on those declared as shareable.

7. **(Previously Presented)** The method according to claim 6, wherein the system is based on a Java Card virtual machine and wherein:

- a program consists of the whole of the code which is found in a "Java Card package";
- the firewall is that of the Java Card Runtime Environment (JCRE);
- the data declared as shareable and therefore sensitive, are objects which are instances of classes which implement the "Javacard.framework.Shareable" interface.

8. **(Previously Presented)** The method according to claim 7, wherein the system stores in sets of sensitive licit references associated with a package all the references which appear in the following cases:

- receiving arguments of "Javacard.framework.Shareable" type when a method of said package is called by another package or by the system,
- "Javacard.framework.Shareable" type return value when said package calls a method from another package or from the system (including a "getAppletSharreableInterfaceObject" method of "Javacard.framework.JCSystem package"),
- reading a public static field of "Javacard.framework.Shareable" type in another package or in the system,

- catching up an instance object of a class from (inheriting from) "java.lang.Throwable" and implementing "Javacard.framework.Shareable".

9. **(Previously Presented)** The method according to claim 1, wherein the whole of the licit stored references is represented by a table.

10. **(Previously Presented)** The method according to claim 1, wherein the set of the licit stored references is emptied, by means of a conservative garbage collector, of references which have become inactive.

11. **(Previously Presented)** The method according to claim 1, wherein:

- the references are represented in the system by handles and tables of pointers,

- the sets of licit stored references are represented by vectors of bits associated with some of the tables of pointers, where a bit has a given index which represents the presence or the absence of the corresponding reference in said sets,

- said vectors of bits are represented by means of a sequence of indexes or lengths corresponding to the extents of bits positioned in the same way.

12. **(Previously Presented)** The method according to claim 1, wherein the references are handles.

13. **(Previously Presented)** The method according to claim 1, wherein the stored licit references are limited to the sole references on data considered as sensitive for the system.

14. **(Previously Presented)** The method according to claim 1, wherein said checks check that the values are among the sensitive licit references

which were stored for this program or else which are references determined as valid and dealing with data which are not sensitive.

15. **(Previously Presented)** The method according to claim 7, wherein the data declared as shareable and therefore sensitive, are objects with public use of the system: global arrays and Entry Point Objects of JCRE.

16. **(Previously Presented)** The method according to claim 11, wherein said vectors of bits are hollow.

17. **(Previously Presented)** The method according to claim 11, wherein some of the tables of pointers are reserved for licit references.